ซอฟต์แวร์
ดออ เพิ่มสมบัติ

เริ่มต้นการเขียนโปรแกรม กราฟิก อย่างมีระบบ (1)

แนะนำวิธีการต่าง ๆ ในการเริ่มต้นการเขียนโปรแกรมกราฟิกอย่างเป็นระบบ

ถ้าคุณอยากเขียนโปรแกรมกราฟิกด้วยเทอร์โบซีรุ่น 1.5 หรือรุ่นที่สูงกว่า และ BGI (Borland Graphics Interface) โปรดตามเรามา และสนุกสนานกับเรา ไม่ว่าคุณจะเป็นมือเก่าหรือมือใหม่ บทความต่อไปนี้จะเน้นถึงการออกแบบทูลส์ (tools) และการประยุกต์ใช้ รวมทั้งการนำเสนอภาพกราฟิก ไอคอนเอดิเตอร์ การระบายสี โปรแกรมทางด้าน CAD และโปรแกรมการเขียนกราฟิกแบบ 3 มิติ

เตรียมอุปกรณ์ให้พร้อม

ควรมีเทอร์โบซี 2.0 คอมพิวเตอร์ IBM PC XT, AT, PS/2 หรือคอมพิวเตอร์ที่ใช้แทนกันได้ และสามารถแสดงผลกราฟิก

โปรแกรมที่เขียนต่อไปนี้ออกแบบสำหรับใช้กับ color Graphics Adapter (CGA) Enhanced Graphics Adaptor (EGA) Virtual Graphics Array (VGA) หรือ Hercules

คุณมาถูกทางแล้ว

ถ้าคุณอ่านบทความนี้เพราะสนใจที่จะเรียนรู้การใช้กราฟิกประยุกต์ในเทอร์โบซีละก็ คุณมาถูกทางแล้ว แต่ก่อนการโปรแกรมทางด้านกราฟิกยากและเป็นงานที่ต้องใช้เวลา โชคดีที่บอร์แลนออกเทอร์โบซี 1.5 และ 2.0 โดยมีเครื่องมือ (tools) ทางด้านกราฟิก ซึ่งเรียกกันว่า Borland Graphics Interface ทำให้งานสร้างกราฟิกแบบ 2 มิติและ 3 มิติโดยใช้คอมพิวเตอร์ส่วนบุคคลได้ง่ายขึ้น

เริ่มหาขั้นตอน

การเขียนโปรแกรมทุกโปรแกรมจะต้องเปิดสู่ภาวะกราฟิกและคืนกลับภาวะข้อความ หลังจากจบโปรแกรมแต่ละโปรแกรมแล้ว ขั้นตอนต่าง ๆ ในการเขียนโปรแกรมกราฟิก คือ ก่อนอื่นจะต้องแน่ใจว่าในเครื่องคอมพิวเตอร์มีแผ่นวงจรที่สามารถจะแสดงกราฟิกได้และที่สับสนก็คือ ในแต่ละเครื่องมีแผ่นวงจรต่าง ๆ กัน แต่ BGI ได้แก้ปัญหาด้วยการสนับสนุนแผ่นวงจรทุกชนิด และเลือกแผ่นวงจรที่ให้รายละเอียดสูงสุด โดยใช้ฟังก์ชัน initgraph( ) ซึ่งมีต้นแบบดังนี้

```
void far initgraph (int far *gdriver,
                    int far *gmode,
                    char far *driver_path);
```

และอยู่ในไฟล์ graphics.h ซึ่งต้องรวมไว้ในตอนต้นโปรแกรมของคุณทุกครั้งที่เขียนโปรแกรมกราฟิก

ไฟล์ graphics.h จะมีโครงสร้างข้อมูล ค่าคงที่ นิยามฟังก์ชันต้นแบบที่ใช้ใน BGI อยู่

ขอให้สังเกตฟังก์ชันต้นแบบ initgraph( ) มี 2 อาร์กิวเมนต์แรกเป็นพอยน์เตอร์ชี้ไปยังจำนวนเต็ม ซึ่งจำนวน-

เต็มนี้จะเป็นค่าทางตรรกะที่บ่งชี้ video adapter และโหมดที่จะใช้ตามลำดับ ส่วนอาร์กิวเมนต์ตัวที่ 3 ชี้ถึงชื่อของพาธที่เก็บไฟล์ไดรเวอร์กราฟิกของเทอร์โบซี ไฟล์ไดรเวอร์เป็นไฟล์ที่ติดมากับแผ่นดิสค์และมีนามสกุลเป็น .bgi ชื่อของพาธไปยังไฟล์ไดรเวอร์เขียนเต็มได้ดังนี้

```
initgraph (&driver, &mode,
"\ \ compilers \ \ turboc \ \ graphics");
```

หรือจะเขียนชื่อพาธเข้าไปด้วยดังนี้

```
initgraph (&driver, &mode,
"graphics \ \ drivers");
```

ประโยคที่สอง initgraph( ) จะค้นหาไดเรคทอรีชื่อไดรเวอร์ซึ่งเป็นไดเรคทอรีย่อยของไดเรคทอรี graphics (graphics เป็นไดเรคทอรีย่อยของไดเรคทอรีปัจจุบันอีกทีหนึ่ง) ถ้าไดรเวอร์กราฟิก BGI เก็บอยู่ในโปรแกรมที่กำลังทำงานอยู่ เราอาจจะเรียก initgraph( ) ได้ดังนี้

```
initgraph (&driver, &mode, " ");
```

ในที่นี้ชื่อพาธ (path) ถูกระบุไว้ว่าเป็นสตริง null ข้อสังเกตคือ เมื่อบ่งชี้ชื่อพาธ คุณต้องใช้ \ สองอัน (เพราะถ้าใช้อันเดียวจะแทนอักษรพิเศษในเทอร์โบซี) ลองมาศึกษาวิธีการใช้ initgraph( ) ในโหมดกราฟิก initgraph( ) ใช้กับไดรเวอร์และโหมดกราฟิกโดยเฉพาะโดยอาร์กิวเมนต์ 2 ตัวแรกจะกำหนดให้เป็นค่าเฉพาะ ทางเลือกอีกอันหนึ่งของการเข้าสู่โหมดกราฟิกคือ ให้ฟังก์ชัน initgraph( ) เลือกโหมดเองโดยอัตโนมัติ และให้พอยน์เตอร์ตัวแรกชี้ไปที่ DETECT แล้ว initgraph( ) จะเลือกโหมดที่มีรายละเอียดสูงสุด ค่าคงที่ DETECT ถูกประกาศไว้ในไฟล์ graphics.h และส่งสัญญาณให้ BGI เลือกโหมดเอง ส่วนอาร์กิวเมนต์ตัวที่สองใน initgraph( ) ไม่จำเป็นต้องกำหนดค่าเริ่มต้นใด ๆ ให้ แต่คุณจำเป็นต้องรวมเอาพอยน์เตอร์ที่ชี้ไปยังจำนวนเต็มเพื่อจะเก็บค่าของอาร์กิวเมนต์ที่ส่งค่าคืนกลับ

เมื่อเราเรียกใช้ initgraph( ) เราอาจจะตรวจสอบค่าของอาร์กิวเมนต์นี้โดยดูว่า BGI เลือกโหมดกราฟิกหรือยัง

จำไว้ว่าการเขียนโปรแกรมกราฟิกด้วย initgraph( ) จะไม่สมบูรณ์ถ้าไม่ปิดโหมดกราฟิกด้วยฟังก์ชัน closegraph( ) ซึ่งไม่ต้องมีอาร์กิวเมนต์และค่าที่คืนกลับ

เขียนโปรแกรม BGI พื้นฐาน

เพื่อสรุปขั้นตอนที่ได้บอกไว้ แล้วลองมาเขียนโปรแกรมที่ต้องใช้ initgraph( ) และ closegraph( ) โปรแกรมต่อไปนี้เป็นโปรแกรมกราฟิกที่เขียนด้วยเทอร์โบซี แสดงให้เห็นถึงโครงสร้างของโปรแกรมกราฟิกซึ่งคุณใช้เป็นต้นแบบในการเขียนโปรแกรมกราฟิกได้ดังโปรแกรม short.c

ถ้าคุณรันโปรแกรมโดยไม่ใส่คำสั่งทางด้านกราฟิกเลย จะเห็นการกะพริบอย่างรวดเร็ว นั่นแสดงว่าคุณเข้าภาวะกราฟิกแล้ว (แต่ในโปรแกรม short.c เป็นการเขียนวงกลมกลางจอภาพหนึ่งวง) จอภาพกะพริบเพราะคอมพิวเตอร์เปลี่ยนจากโหมดข้อความปกติ (text) ไปเป็นโหมดกราฟิก แล้วกลับไปโหมดข้อความปกติอีกที

```
#include<graphics.h>
main()
{
  int gdriver=DETECT;
  int gmode;
  int xmax,ymax;

  initgraph(&gdriver,&gmode,"");
  /*put drawing command here...*/
  xmax=getmaxx();
  ymax=getmaxy();
  circle(xmax/2,ymax/2,30);

  getch();
  closegraph();
}
```

รูปที่ 1
โปรแกรม short.C

ลองพิจารณาโปรแกรมโดยละเอียดกันดีกว่า โปรแกรมเริ่มด้วยถ้อยแถลง #include สำหรับไฟล์เฮดเดอร์ graphics.h คงจำได้ว่าโปรแกรมทางด้านกราฟิกจะต้องมีถ้อยแถลง

```
#include<graphics.h>
```

เสมอ

การเรียก initgraph( ) จะเป็นการเซตระบบ video adapter เพราะว่า DETECT ถูกส่งผ่านเป็นค่าแรกของอาร์กิวเมนต์ ส่วนพารามิเตอร์ตัวที่สองซึ่งอ้างถึงโหมด video ไม่ได้ให้ค่าอะไร

อาร์กิวเมนต์ 2 ตัวแรกถูกส่งผ่านเป็นแบบพอยน์เตอร์ไปยังค่าของจำนวนเต็ม เมื่อเรียกใช้ initgraph( ) คุณอาจสำรวจ gdriver และ gmode เพื่อจะดูว่าวิดิโออะแดปเตอร์ (video adapter) ตัวไหนที่ถูกโหลด และจะให้ค่าเริ่มต้นแก่โหมดใด ส่วนอาร์กิวเมนต์ตัวที่สามบอกว่า BGI อยู่ที่ไดเรคทอรีใด

ถ้าคุณบอกไดเรคทอรีไม่ถูกต้อง โปรแกรมของคุณจะล้มเหลว ในการจะกระทำเกี่ยวกับโหมดกราฟิกจึงควร

เขียนโปรแกรมตรวจสอบการผิดพลาดตอนที่โหลดไฟล์ไดรเวอร์ด้วยจะเป็นการดี

ฟังก์ชันที่ใช้ในการตรวจสอบคือ

```
graphsesult ( )
```

จะให้ค่า error code สำหรับการทำงานกราฟิกคำสั่งสุดท้าย

```
grapherrormsg( )
```

ให้ข้อความบ่งบอกความผิดพลาดตาม error code'

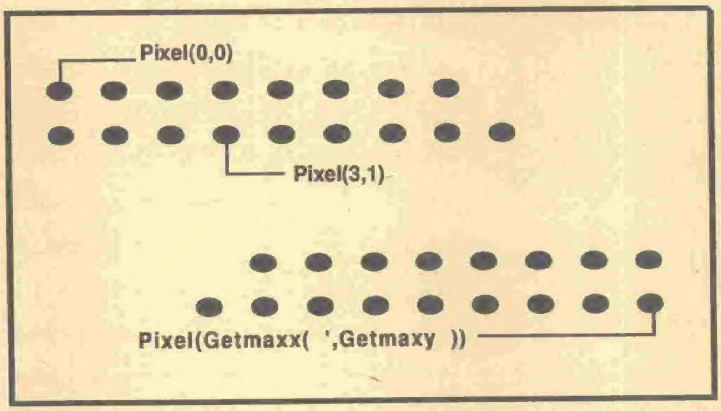
เราจะใช้ฟังก์ชันตรวจสอบความผิดพลาดตามโปรแกรม errtest.c

มองดูการตรวจสอบข้อผิดพลาด

ข้อผิดพลาดปกติที่มักจะเกิดขึ้นคือ ไม่พบไดรเวอร์กราฟิกจาก initgraph( ) อาร์กิวเมนต์ตัวที่สามจะชี้พาธของไฟล์ไดรเวอร์กราฟิกในเทอร์โบซี ถ้าพาธไม่ถูกต้องก็จะไม่เจอไดรเวอร์กราฟิก การกำหนดค่าเริ่มต้นให้แก่กราฟิกก็ไม่เกิดขึ้น

ถ้าโปรแกรมถูกต้อง ผลลัพธ์จากการรันโปรแกรมคือ

```
Hello graphics world! Press any
key to exit...
```



รูปที่ 3

แต่ถ้าเกิดปัญหาขึ้น ก็แสดงว่ามีข้อผิดพลาดให้เห็นและโปรแกรมจะหยุดชะงัก (terminate) เพื่อความแน่ใจในสิ่งแวดล้อมด้านกราฟิกของคอมพิวเตอร์ คุณควรจะทดสอบโปรแกรม errtest.c ก่อนที่จะเขียนโปรแกรมกราฟิกใด ๆ

ฟังก์ชันใหม่ที่พบกันในโปรแกรม errtest.c คือ

```
Outtext( )
```

คือการเขียนข้อความลงจอ เริ่มที่ตำแหน่งที่เคอร์เซอร์อยู่ ซึ่งต่อไปคุณจะพบฟังก์ชัน outtextxy( ) เป็นการเขียนข้อความลงจอโดยเริ่มตามตำแหน่งที่กำหนดในบทความเกี่ยวกับฟอนต์ (fonts) และข้อความ (text) ตอนต่อไป

การทำงานเกี่ยวกับโคออร์ดิเนต

งานทุกชิ้นจะทำได้เสร็จตามเป้าหมายหรือไม่ คุณต้องรู้ว่าขณะนี้คุณกำลังทำอะไรและจะมุ่งไปทางไหนเพื่อบรรลุเป้าหมาย ในการเขียนโปรแกรมกราฟิกก็เช่นกัน จะต้องรู้ว่าโคออร์ดิเนตของจุดภาพ (pixel) อยู่ที่ใดบนจอ (pixel ย่อมาจาก picture elements เป็นจุดเล็ก ๆ บนจอภาพ) การบอกตำแหน่งของจุดภาพจะใช้ระบบโคออร์ดิเนตเหมือนกับวิธีที่เข้าถึงแถวและหลักในโครงสร้างข้อมูลของอะเรย์แบบ 2 มิติ

การขยายโคออร์ดิเนตของระบบขึ้นอยู่กับ video adapter ซึ่งคุณใช้อยู่ แต่โชคยังดีที่มีเทคนิคมาตรฐานสำหรับจัดการกับโคออร์ดิเนตของพิกเซลที่คุณสามารถใช้ได้กับ BGI ทุกตัวที่สนับสนุน (support) video adapter

โหมดของกราฟิกในคอมพิวเตอร์ส่วนบุคคล เริ่มโคออร์ดิเนตแรกที่มุมบนซ้ายของจอภาพด้วยโคออร์ดิเนต (0,0) โคออร์ดิเนตของพิกเซลที่มีค่าสูงสุดจะอยู่ล่างขวาของจอภาพ และโคออร์ดิเนตขึ้นอยู่กับโหมดกราฟิกที่คุณมี เทอร์โบซีจึงมีสองฟังก์ชันที่ใช้เรียก

```
/*errtest.c Test program how to use graphics driver*/
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
main()
{
   int gdriver=DETECT;
   int gmode,gerror;

   initgraph(&gdriver,&gmode,"");
   gerror=graphresult();
   if(gerror(0){
    printf("Graphics initialization error:");
    printf("%s\n",grapherrormsg(gerror));
    exit(1);
   }
   outtext("Hello graphics world!    Press any to exit...");
   getch();
   closegraph();
   return(0);
   }
```

รูปที่ 2
โปรแกรม errtest.C

พิกเซลที่อยู่มุมขวาล่างดังนี้

```
maxx = getmaxx( );
maxy = getmaxy( );
```

getmaxx( ); ให้ค่าพิกัดสูงสุดในแกน X ปัจจุบัน
getmaxy( ); ให้ค่าพิกัดสูงสุดในแกน y ปัจจุบัน

คำสั่งที่ใช้วาดรูป

เทอร์โบซีมีคำสั่งที่ใช้วาดรูปอยู่มากมาย และแต่ละคำสั่งก็ง่ายต่อการใช้ คำสั่งเหล่านี้เริ่มจากระดับพิกเซลบนจอภาพไปจนถึงฟังก์ชันระดับสูงที่ใช้วาดแผนภูมิแท่ง 3 มิติทีเดียว

เราจะสำรวจฟังก์ชันที่ใช้วาดรูปต่าง ๆ โดยเริ่มจากพิกเซลก่อน แล้วค่อยขยับไปสู่ฟังก์ชันที่สูงขึ้น putpixel( ) จะลงจุดที่ตำแหน่ง X, Y ด้วยสีเฉพาะ getpixel( ) ให้ค่าสีของ pixel ที่ตำแหน่ง X,Y ทั้งสองฟังก์ชันนี้จะเข้าถึง 1 pixel ในเวลาใดเวลาหนึ่งเท่านั้น ฟังก์ชันต้นแบบคือ

```
unsigned far getpixel (int x, int y);
void far putpixel (int x, int y, int color);
```

ฟังก์ชัน getpixel( ) จะคืนค่าที่สอดคล้องกับสีของพิกเซล ณ ตำแหน่ง x,y การเรียกใช้ฟังก์ชันจะทำได้ดังนี้ เช่น

```
putpixel(150,200,BLUE);
color = getpixel (30,60);
```

โปรแกรม randpixe.c แสดงวิธีการใช้ putpixel( ) และ getpixel( ) โดยการสุ่มจุดภาพขึ้นมา 1,000 pixels บนจอภาพ แล้วลงจุดด้วย putpixel( ) แล้วใช้ getpixel( ) เปลี่ยนสี ทำให้จุดเกิดบนจอภาพค่อย ๆ หายไป

โปรแกรมเริ่มด้วยการใช้ฟังก์ชันตรวจสอบโหมดกราฟิกเพื่อตรวจสอบจอ แล้วจึงส่งผ่านค่าให้ฟังก์ชัน initgraph( ) (การระบุ ถ้าเป็นกราฟไดรเวอร์ควรจะระบุโดยใช้ค่าคงที่ที่กำหนดไว้ก่อนแล้วใน graphics.h ซึ่งมีค่าต่าง ๆ ดังรูปที่ 1) ในที่นี้เราให้กราฟไดรเวอร์ (gdrive) มีค่าเป็น DETECT แสดงว่าฟังก์ชัน initgraph( ) จะเป็นการเลือกจอเอง เมื่อเซตเป็นโหมดกราฟิกแล้วก็จะเรียกใช้ค่า x และ y ที่มีค่าสูงสุดด้วยฟังก์ชัน getmaxx( ) และ getmaxy( ) และในขณะเดียวกันก็เลือกค่าสูงสุดของสีด้วยฟังก์ชัน getmaxcolor( ) (getmaxcolor( ) เป็นฟังก์ชันที่ให้ค่าสูงสุดของสีที่วาดได้ในโหมดปัจจุบัน)

ฟังก์ชัน randomize( ) เป็นฟังก์ชันที่จะใช้สุ่มตัวเลข โดยที่ตัวเลขแต่ละตัวในกลุ่มมีโอกาสที่จะถูกเลือกเท่า ๆ กัน และมีฟังก์ชันต้นแบบอยู่ในไฟล์เฮดเดอร์ stalib.h

```
#include<stalib.h>
int random(int num);
```

ซึ่งให้ค่าคืนกลับเป็นเลขที่อยู่ระหว่าง 0 กับ 1 ดูรายละเอียดใน USER'S Guide Reference ของ Turbo C

การวาดรูป

เราได้เรียนและเห็นการลงจุดหรือการอ่านพิกเซลมาแล้ว คราวนี้เราจะใช้ฟังก์ชันที่มีอยู่ใน BGI ของเทอร์โบซีมาลองวาดรูปกันดีกว่า โดยเริ่มใช้ฟังก์ชันต่อไปนี้

- arc( ) วาดส่วนของวงกลม
- circle( ) วาดวงกลม
- drawpoly( ) วาดรูปหลายเหลี่ยม
- ellipse( ) วาดวงรีหรือส่วนของวงรี
- line( ) วาดเส้นโดยกำหนดพิกัดแรกและสุดท้าย
- rectangle( ) วาดรูปสี่เหลี่ยม

แต่ละฟังก์ชันจะมีลักษณะเฉพาะตัวของมัน เช่น ฟังก์ชันที่ใช้ในการเขียนวงกลมจะมีอาร์กิวเมนต์ 3 ตัว คือ อาร์กิวเมนต์ตัวแรก (x) อาร์กิวเมนต์ตัวที่สอง (y) แทนพิกัดของจุดศูนย์กลางวงกลม ส่วนอาร์กิวเมนต์ตัวที่ 3 จะแทนรัศมี (r) เช่น จะวาดวงกลมที่มีจุดศูนย์กลางที่ (100,100) และรัศมี 80 ก็จะเรียกใช้ฟังก์ชันดังนี้

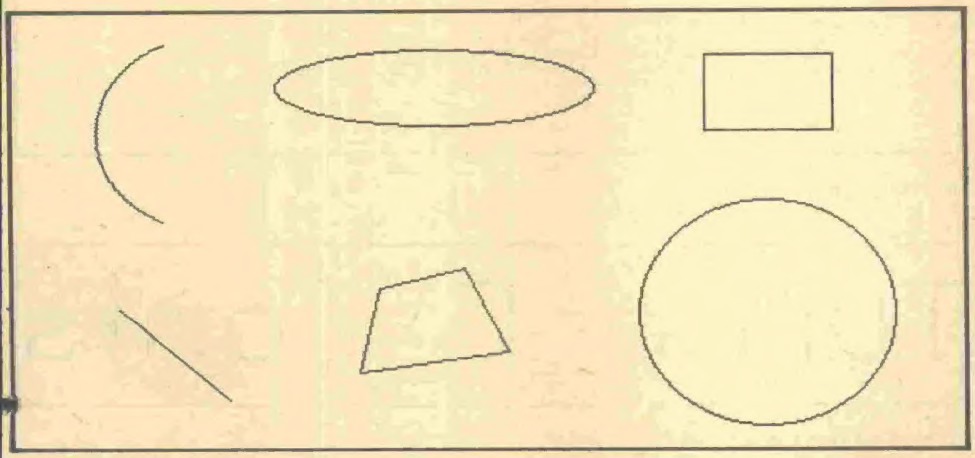
```
circle (100,100,80);
```

```
#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
main()
{
  int gdriver=DETECT,gmode;
  int maxx,maxy;
  int maxcolor;
  int i,x,y,color;

  initgraph(&gdriver,&gmode,"");
  randomize();
  maxx=getmaxx();
  maxy=getmaxy();
  maxcolor=getmaxcolor();
  for (i=0;i<1000;i++){
   x=random(maxx+1);
   y=random(maxy+1);
   color=random(maxcolor+1);
   putpixel(x,y,color);
  }
  i=0;
  for(y=0;y<=maxy;y++)
    for(x=0;x<=maxx;x++)
      if(getpixel(x,y) !=BLACK) {
       if(i%2==0)
         putpixel(x,y,BLACK);
         i++;
       }
       outtextxy(0,0,"Press any Key to Continue...");
       getch();
       closegraph();
       return(0);
       }
```

รูปที่ 4
โปรแกรม randpixe.C

```
#include<graphics.h>
#include<conio.h>
main()
{
  int gdriver=DETECT;
  int gmode,maxx,maxy;
  int points[10];

  initgraph(&gdriver,&gmode," ");
  maxx=getmaxx();
  maxy=getmaxy();
  arc(maxx/6,maxy/4,0,135,50);
  outtextxy(maxx/6-textwidth("arc")/2,0,"arc");
  circle(maxx/2,maxy/4,60);
  outtextxy(maxx/2-textwidth("circle")/2,0,"circle");
  points[0]=maxx*5/6-20;
  points[1]=maxy/4-20;
  points[2]=maxx*5/6-30;
  points[3]=maxy/4+25;
  points[4]=maxx*5/6+40;
  points[5]=maxy/4+15;
  points[6]=maxx*5/6+20;
  points[7]=maxy/4-30;
  points[8]=points[0];
  points[9]=points[1];
  drawpoly(5,points);
  outtextxy(maxx*5/6-textwidth("drawpoly")/2,0,"drawpoly");
  ellipse(maxx/6,maxy*3/4,0,360,75,20);
  outtextxy(maxx/6-textwidth("ellipse")/2,maxy-textheight("1"),
            "ellipse");
  line(maxx/2-25,maxy*3/4-25,maxx/2+25,maxy*3/4+25);
  outtextxy(maxx/2-textwidth("line")/2,maxy-textheight("1"),"line");
  rectangle(maxx*5/6-30,maxy*3/4-20,maxx*5/6+30,maxy*3/4+20);
  outtextxy(maxx*5/6-textwidth("rectangle")/2,maxy-textheight("1"),"rectangle")
  getch();
  closegraph();
  return(0);
}
```

รูปที่ 5 โปรแกรม draw.C



รูปที่ 6 รูปของโปรแกรม draw.C

ส่วนฟังก์ชันอื่น ๆ ก็มีลักษณะคล้าย ๆ กัน จะแปลกออกไปก็คือ ฟังก์ชัน drawpoly( ) จะรับเอาอะเรย์ของจุด x และ y และจำนวนจุดที่อยู่ในอะเรย์ แล้วพล็อตส่วนของเส้นตรงเชื่อมจุดเหล่านั้น ไปจนกระทั่งจุดแรกและจุดสุดท้ายพบกันจึงเกิดเป็นรูปหลายเหลี่ยมขึ้น เราลองมาเขียนโปรแกรมแล้วดูผลลัพธ์ก่อน ส่วนรายละเอียดจะกล่าวต่อไปในตอนหลัง

โปรแกรมที่เขียนขึ้นจะอยู่ในไฟล์ draw.c โดยจะแบ่งจอภาพเป็น 6 ส่วน แล้วใช้ฟังก์ชัน outtextxy บอกว่าภาพที่วาดขึ้นมีชื่อว่าอะไร ฟังก์ชันต่าง ๆ ที่แสดงรายละเอียดของอาร์กิวเมนต์ในการวาดรูปมีรายละเอียดดังนี้

```
arc (x,y,st_angle,end_angle,radius);
circle (x,y,radius);
drawpoly (number_of_points,array_of_
xy_points);
ellipse (x,y,st_angle,end_angle,xradius,
yradius);
line (x1,y1,x2,y2);
rectangle (left,top,right,bottom);
```

จากโปรแกรมเราใช้ textheight( ) และ textwidth( ) เพื่อจะหาขนาดของอักษรที่แสดงบนจอ textheight( ) ให้ค่าความสูงของข้อความเป็นพิกเซล textwidth( ) ให้ค่าความกว้างข้อความเป็นพิกเซล

การระบายรูป

ในตอนต้นเรากล่าวถึงการสร้างรูปต่าง ๆ ต่อไปนี้จะลองระบายรูปนั้น ๆ โดยใช้ฟังก์ชัน bar( ), bar3d( ), fillpoly( ), fillellipse( ), pieslice( ) และ sector( )

– bar( )-วาดรูปกราฟแท่งและระบายสีและแบบลงบนกราฟ

– bar3d( ) วาดรูปกราฟแท่ง 3 มิติและทำการระบายสีและแบบลงบนกราฟ

– fillpoly( ) ระบายสีและแบบลงในรูปหลายเหลี่ยม

– fillellipse( )

– pieslice( ) วาดรูปส่วนของ pie พร้อมระบายสีและแบบ

– sector( )

ฟังก์ชันต้นแบบในการระบายและให้สีจะมีดังนี้

– void far bar (int left, int top, int right, int bottom);

– void far bar3d (int left, int top, int right, int bottom, int depth, int top_flag);

– void far fillpoly int number of points, int far *array_of_xy_

ponts);

– void far fillellipse (int x, int y, int xradius, int yradius);

– void far pieslice (int x, int y, ing start__tangle, int end__angle, int radius);

– void far sector (int x, int y, int start__angle, int end__argle, int xradius, int yradius);

คุณคงสงสัยใช่ไหมว่า คำสั่งที่ให้ระบายจะรู้ได้อย่างไรว่าจะใช้รูปแบบใดในการระบาย โดยพื้นฐานเวลาที่คุณเรียกฟังก์ชันให้ระบาย มันจะใช้สีและรูปแบบปัจจุบันในการระบาย ในโปรแกรมของเรา (showfill.c) เราไม่ได้บอกรูปแบบของการระบาย โปรแกรมจะเลือกเองโดยการระบายทับในสีสูงสุดที่มีอยู่ใน video adaptor ซึ่งเทอร์โบซีจะมีฟังก์ชัน setfillstyle( ) (เลือกลักษณะการระบาย) ซึ่งถูกใช้ในการเปลี่ยนรูปแบบได้ตลอดเวลา เช่น

```
setfillityle (HATCH__FILL, BLUE);
```

ก็จะระบายเป็นรูปตาข่ายสีน้ำเงิน (ระบบ EGA) ในตอนต่อไปจะพูดถึงวิธีการที่ผู้เขียนโปรแกรมเลือกสีและรูปแบบได้ตามใจชอบ

ข้อความและฟอนต์

เมื่อกล่าวถึงผลลัพธ์ที่เป็นข้อความและฟอนต์ BGI ให้ผลลัพธ์ที่ประทับใจมากทีเดียว กล่าวคือ โหมดกราฟิกที่มีอยู่บนเครื่องไอบีเอ็มพีซีสนับสนุนในด้านอักษรที่มีขนาดเฉพาะ ซึ่งบางครั้งมีคุณภาพต่ำ แต่เครื่องมือ BGI จะเพิ่มข้อความ (text) ให้มีคุณภาพดีขึ้นได้ถ้าต้องการแสดงผลในรูปของกราฟิก

BGI ยังรวมเอาแพคเกจ stroke font ซึ่งเหมาะที่จะเขียน (วาด) อักขระที่มีความสูงและความกว้างต่าง ๆ กัน และยังสามารถเขียนในแนวตั้งได้อีกด้วย

งานต่อมาเราจะลองเขียนโปรแกรมเพื่อแสดงข้อความ 1 บรรทัดที่มีสเกลทั้งทางด้านแนวนอนและแนวตั้งต่าง ๆ กัน โปรแกรมจะใช้ san serif stroke font เพื่อแสดงข้อความ "Turbo C Font" บนกลางจอภาพ โปรแกรมยังให้คุณเปลี่ยนแปลงสเกลของขนาดของข้อความได้อีกด้วยโดยที่ถ้าคุณกด

s จะทำให้ข้อความย่นย่อขนาด

g จะทำให้ข้อความขยายขนาด

space bar จะทำให้ข้อความเปลี่ยนจากแนวนอนเป็นแนวตั้ง

q ออกจากโปรแกรม

```
/*showtext.c---This program displays aline of text that you can
 *scale as you please.The programs allows the following user
 *interaction
 *        s    -shrink the test
 *        g    -grow or enlarge tue text
 *        space-switch between horizontal and vertical text format
 *        q    -quit the program
 */
#include<graphics.h>
#include<conio.h>
#define INC 10
main()

{
 int gdriver=DETECT,gmode;
 char ch;
 char string[]="Turbo C Fonts";
 int cx,cy,sw,sh;
 int multx,divx,multy,divy,dir;
 initgraph(&gdriver,&gmode,"");
 setfillstyle(SOLID_FILL,BLACK);
 cx=getmaxx()/2; cy=getmaxy()/2;
 multx=100;        divx=100;
 multy=100;        divy=100;
 dir=HORIZ_DIR;
 settextjustify(CENTER_TEXT,CENTER_TEXT);
 setusercharsize(multx,divx,multy,divy);
 settextstyle(SANS_SERIF_FONT,dir,USER_CHAR_SIZE);
 outtextxy(cx,cy,string);
 while((ch=getch()) !='q'){
 sw=textwidth(string)/2;
 sh=textheight(string)*3/2;
 if(dir==HORIZ_DIR){
 if(sw>cx) sw=cx+1;
 if(sh>cy) sh=cy+1;
 bar(cx-sw,cy-sh,cx+sw,cy+sh);
 }
 else {
   if (sh>cx) sh=cx+1;
   if(sw>cy) sw=cy+1;
   bar(cx-sh,cy-sw,cx+sh,cy+sw);
  }
  if(ch=='g'){
   if(divx>INC){
     divx-=INC;multx +=INC;
    }
    if(divy>INC){
    divy-=INC;multy+=INC;
    }
   }
   else if (ch=='s'){
    if(multx >INC){
    multx-=INC;divx +=INC;
    }
   }
   else if (ch=='')
     dir=(dir==HORIZ_DIR)? VERT_DIR :HORIZ_DIR;
   setusercharsize(multx,divx,multy,divy);
   settextstyle(SANS_SERIF_FONT,dir,USER_CHAR_SIZE);
   outtextxy(cx,cy,string);
 }
 closegraph();
 return(0);
}
```

รูปที่ 7 โปรแกรม showtext.C

รูปที่ 8 ผลลัพธ์ของ showtext.C